1-1-1981

Algebraic Complexity Theory

Nicholas Pippenger
Harvey Mudd College

Nicholas Pippenger

Algebraic Complexity Theory

Algebraic complexity theory, the study of the minimum number of operations sufficient to perform algebraic computations, is surveyed with emphasis on the general theory of bilinear forms and two of its applications: polynomial multiplication and matrix multiplication. Though by no means exhausting algebraic complexity theory, these topics illustrate well its development and its methods, and provide examples of its most striking successes.

Introduction

Algebraic complexity theory is the study of the minimum number of operations sufficient to perform various computations, in cases where these computations are of an algebraic nature. To begin with a concrete example, suppose that we are given the real and imaginary parts of two complex numbers, $a + bi$ and $c + di$, and that we wish to compute the real and imaginary parts of their product, $e + fi$. These may be computed using the formulae $e = ac - bd$ and $f = ad + bc$, which require four multiplications, one addition, and one subtraction. An alternate method, however, is to compute $x = (a + b) \times (c - d)$, $y = ad$, and $z = bc$, then to compute $e = x + y - z$ and $f = y + z$. This method requires three additions and two subtractions, but only three multiplications. Thus, if addition and subtraction take much less time than multiplication (as indeed they do on many computing machines), the alternate method may be faster than the original one.

Encouraged by this success, one may ask if there is an algorithm for complex multiplication requiring only two multiplications, together with any number of additions and subtractions. In 1971, Winograd [1] showed that there is not. This is a result on a different scale of significance from that of the preceding paragraph; it calls not for the discovery of a single algorithm for performing the computation but for an analysis of *all* such algorithms. Such an analysis must begin by formulating definitions of what a computation is, what an algorithm is, and what it means for an algorithm to perform a computation.

The following formulation will be used in this paper. We are given certain *input data* $\{B_j\}_{1 \le j \le J}$, which are regarded as indeterminates. (For complex multiplication, these are $\{a, b, c, d\}$.) We wish to compute certain *output data* $\{C_k\}_{1 \le k \le K}$. (For complex multiplication, these are $\{e, f\}$.) If these are to be computed from the input data by means of real constants, additions, subtractions, and multiplications, these will be polynomials in the input data with coefficients in $\mathbf{R}$, the field of real numbers; thus, they are regarded as elements of $\mathbf{R}[\{B_j\}_{1 \le j \le J}]$, the ring of all such polynomials. (For complex multiplication, these are $\{ac - bd, ad + bc\}$.) An *algorithm* for computing $\{C_k\}_{1 \le k \le K}$ from $\{B_j\}_{1 \le j \le J}$ is a sequence of elements of $\mathbf{R}[\{B_j\}_{1 \le j \le J}]$ that contains each element of $\{C_k\}_{1 \le k \le K}$ and in which each element is an element of $\{B_j\}_{1 \le j \le J}$, an element of $\mathbf{R}$, or the sum, difference, or product of two elements preceding it in the sequence. (The sequences

$$a, b, c, d, ac, bd, ac - bd, ad, bc, ad + bc$$

and

$$a, b, c, d, a + b, c - d, ac - ad + bc - bd,$$
$$ad, bc, ac + bc - bd, ac - bd, ad + bc$$

represent the two algorithms for complex multiplication just described.)

In terms of this formulation, Winograd's argument to the effect that complex multiplication cannot be performed with only two multiplications can be sketched

briefly as follows. First, all possible choices for the outcome of the first multiplication, say x, are considered. Clearly, x must be the product of two polynomials of the first degree in a, b, c, and d. Next, a criterion is obtained for the possibility of computing two polynomials, e and f, from a, b, c, d, and x with only one more multiplication. The criterion is that the determinant of a certain 2×2 matrix must vanish. Finally, it is shown that for no choice of x is the criterion satisfied. The proof, though too lengthy to recount in full here, uses only elementary algebraic reasoning. It actually establishes more than what is stated above, for it shows that three multiplications are necessary, even if *scalar* multiplications (multiplications in which one of the factors is an element of **R**) are not counted. Winograd's result can thus be expressed by saying that three *nonscalar* multiplications are required to perform complex multiplication.

The problem of complex multiplication just discussed illustrates the principal theme of algebraic complexity theory. The goal of this paper is to show how this theme has been developed with regard to other more challenging computational problems. The problems to be discussed involve the manipulation of polynomials and matrices; they have been chosen both for their intrinsic mathematical interest and because they advantageously exhibit many of the basic techniques of algebraic complexity theory.

Bilinear forms

This section deals not with a specific computational problem but rather with the general framework within which the problems of the next two sections will be discussed. We suppose that we are given *two* sets of input data, $\{A_i\}_{1 \le i \le I}$ and $\{B_j\}_{1 \le j \le J}$, and that the output data $\{C_k\}_{1 \le k \le K}$ that we wish to compute constitute a set of *bilinear forms* in these input data, *i.e.*, that the output data can be expressed as

$$C_k = \sum_{\substack{1 \le i \le I, \\ 1 \le j \le J}} T_{i,j,k} A_i B_j , \qquad (1)$$

where the coefficients $\{T_{i,j,k}\}_{1 \le i \le I, 1 \le j \le J, 1 \le k \le K}$ belong to the underlying field and characterize the problem to be solved within this class. This class of problems embraces many important ones: Complex multiplication, discussed in the introduction, falls within it (I, J, and K are 2, the As are a and b, the Bs are c and d, the Cs are c and f, and the Ts are all -1, 0, or $+1$), as does quaternion multiplication (see de Groote [2] and a forthcoming paper of Feig for an analysis of quaternion multiplication). More important problems in this class are polynomial multiplication and matrix multiplication, which will be discussed in the following sections.

The $I \times J \times K$ array of coefficients that characterizes a set of bilinear forms is often called a *tensor*. If T is such a tensor, we shall denote by $\mu(T)$ the minimum number of nonscalar multiplications sufficient to compute the bilinear forms (1). Here, additions and subtractions, as well as multiplications in which one of the factors is a constant, are not counted.

It will be noted that no mention was made of division in the preceding paragraph. Ungar (see Winograd [3]) observed that no loss is entailed by this omission. If a set of bilinear forms can be computed by an algorithm using L nonscalar multiplications and divisions (nonscalar divisions are those in which the divisor is not a constant), then it can also be computed by an algorithm using L nonscalar multiplications but no divisions (the number of additions, subtractions, and scalar multiplications may, of course, increase). Thus, throughout the rest of this section divisions will be ignored. (Strassen [4] obtained a noteworthy generalization of Ungar's result to multilinear forms: If a set of d-linear forms can be computed by an algorithm using L nonscalar multiplications and divisions, then it can also be computed by an algorithm using at most $(d-1)L$ nonscalar multiplications but no divisions.)

Let $\mu^*(T)$ denote the minimum number of nonscalar multiplications sufficient to compute the bilinear forms (1) when the indeterminates $\{A_i\}_{1 \le i \le I}$ and $\{B_j\}_{1 \le j \le J}$ are not assumed to commute, *i.e.*, when identities such as $A_i B_j = B_j A_i$ cannot be relied upon to establish the correctness of algorithms. That $\mu^*(T)$ may be larger than $\mu(T)$ can be seen by comparing the results of Hopcroft and Kerr [5], who showed that $\mu^*(T) = \lceil 7N/2 \rceil$ for the problem T of multiplying a 2×2 matrix by a $2 \times N$ matrix, with the results of Winograd [6], who showed that $\mu(T) \le 3N + 2$ for this problem. Winograd [7] has shown, however, that $\mu^*(T) \le 2\mu(T)$ for any problem of computing a set of bilinear forms, and Ja' Ja' [8] has obtained even sharper bounds on the power of commutativity in computing a set of bilinear forms. (Hyafil [9] has shown that for computing multilinear forms—specifically, the determinant of a matrix—the power of commutativity is greater; it can reduce the complexity of a problem from an exponential to a polynomial in the number of input data.)

Once divisions and commutativity have been excluded, a great deal can be said about the structure of optimal algorithms for computing sets of bilinear forms. They are what may be called *bilinear algorithms:* algorithms in which every nonscalar multiplication is the product of a *linear form* in $\{A_i\}_{1 \le i \le I}$ with a linear form in $\{B_j\}_{1 \le j \le J}$,

$$M_l = \left(\sum_{1 \le i \le I} \alpha_{i,l} A_i \right) \left(\sum_{1 \le j \le J} \beta_{j,l} B_j \right) ,$$

and in which the $\{C_k\}_{1\le k\le K}$ are computed as linear forms in $\{M_l\}_{1\le l\le L}$,

$$C_k = \sum_{1\le l\le L} \gamma_{k,l} M_l$$

(here, $\{\alpha_{i,l}\}_{1\le i\le I,1\le l\le L}$, $\{\beta_{j,l}\}_{1\le j\le J,1\le l\le L}$, and $\{\gamma_{k,l}\}_{1\le k\le K,1\le l\le L}$ are coefficients from the underlying field). The minimum number L of nonscalar multiplications in any such algorithm is equal to the minimum number L for which the system of IJK equations

$$T_{i,j,k} = \sum_{1\le l\le L} \alpha_{i,l}\beta_{j,l}\gamma_{k,l} \tag{2}$$

has a solution. This number is called the *rank* $\rho(T)$ of the tensor T, by analogy with the rank of a matrix $\{S_{i,j}\}_{1\le i\le I,1\le j\le J}$, which may be defined as the minimum number L for which the system of IJ equations

$$S_{i,j} = \sum_{1\le l\le L} \alpha_{i,l}\beta_{j,l},$$

has a solution. That this is so was observed independently by Gastinel [10], Fiduccia [11], Strassen [12], and doubtless others. The minimum number L is called the *rank* $\rho(T)$ of the tensor T and $\mu^*(T) = \rho(T)$ for sets of bilinear forms in noncommuting indeterminates.

If there is just one bilinear form to be computed ($K = 1$), the tensor T reduces to a matrix, the rank of which is easily determined by standard methods of linear algebra. For two bilinear forms ($K = 2$), the problem is more difficult, but a complete solution has been obtained by Grigorev [13] and Ja' Ja' [14, 15]. But in general ($K \ge 3$), no satisfactory method is known for determining the rank of a given tensor. (If the entries of the tensor are integers and the underlying field is real or complex, the rank can be computed by a general decision procedure for the first-order theory of real-closed fields, but such a method is infeasible even for quite small tensors.)

One consequence of the identity $\mu^*(T) = \rho(T)$ follows immediately from the symmetry among $\alpha_{i,l}$, $\beta_{j,l}$, and $\gamma_{k,l}$ in (2): The rank of a tensor, and thus the minimum number of nonscalar multiplications sufficient to compute the associated set of bilinear forms, is the same for all six transpositions of a tensor obtained by permuting its three coordinate axes. This principle is usually referred to as "duality."

We close this section by mentioning an outstanding open problem. Suppose that in addition to computing the bilinear forms $\{C_k\}_{1\le k\le K}$ given by a tensor T from the indeterminates $\{A_i\}_{1\le i\le I}$ and $\{B_j\}_{1\le j\le J}$ we wish to compute another set of bilinear forms $\{C'_k\}_{1\le k\le K'}$ given by a tensor T' from the indeterminates $\{A'_i\}_{1\le i\le I'}$ and $\{B'_j\}_{1\le j\le J'}$, which are assumed to be disjoint from $\{A_i\}_{1\le i\le I}$ and $\{B_j\}_{1\le j\le J}$. If $T \oplus T'$ (the *direct sum* of T and T') denotes the tensor of $\{C_k\}_{1\le k\le K} \cup \{C'_k\}_{1\le k\le K'}$ as bilinear forms in $\{A_i\}_{1\le i\le I} \cup \{A'_i\}_{1\le i\le I'}$ and $\{B_j\}_{1\le j\le J} \cup \{B'_j\}_{1\le j\le J'}$, then $\mu^*(T \oplus T') \le \mu^*(T) + \mu^*(T')$, since one may combine optimal algorithms for T and T' into an algorithm for $T \oplus T'$. The *direct sum conjecture*, due to Strassen [4], is that $\mu^*(T \oplus T') = \mu^*(T) + \mu^*(T')$, *i.e.*, that disjoint problems may as well be solved separately.

Polynomial multiplication

The problem of algebraic complexity theory that has, on the one hand, enjoyed the most dramatic reduction in the number of operations required and, on the other, bestowed this reduction on the most numerous and varied applications is that of polynomial multiplication or convolution. Suppose that we are given two polynomials $A(x)$ (of degree at most I) and $B(x)$ (of degree at most J) by means of their coefficients $\{A_i\}_{0\le i\le I}$ and $\{B_j\}_{0\le j\le J}$ (A_i is the coefficient of x^i, and B_j the coefficient of x^j) and that we wish to compute the coefficients $\{C_k\}_{0\le k\le K}$ of the product $C(x)$ (of degree at most $K = I + J$). These output data are given by the formulae

$$C_k = \sum_{i+j=k} A_i B_j, \tag{3}$$

which reveal them to be a set of bilinear forms in the input data.

An obvious algorithm based on the formulae (3) requires $(I + 1)(J + 1)$ multiplications and IJ additions. A large class of alternate algorithms derives from the following strategy. First, choose $K + 1$ distinct points $\{\xi_k\}_{0\le k\le K}$. Second, compute $\{A(\xi_k)\}_{0\le k\le K}$ from $\{A_i\}_{0\le i\le I}$, and $\{B(\xi_k)\}_{0\le k\le K}$ from $\{B_j\}_{0\le j\le J}$ [evaluate $A(x)$ and $B(x)$ at the points $\{\xi_k\}_{0\le k\le K}$]. Third, compute $\{C(\xi_k)\}_{0\le k\le K}$ by means of the formulae

$$C(\xi_k) = A(\xi_k)B(\xi_k). \tag{4}$$

Fourth, compute $\{C_k\}_{0\le k\le K}$ from $\{C(\xi_k)\}_{0\le k\le K}$ [interpolate $C(x)$ through the points $\{\xi_k\}_{0\le k\le K}$].

The second step of this strategy can be expressed as the multiplication of a matrix by two vectors:

$$\begin{bmatrix} A(\xi_0) \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ A(\xi_K) \end{bmatrix} = \begin{bmatrix} \xi_0^0 & \cdots & \xi_0^K \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \xi_K^0 & \cdots & \xi_K^K \end{bmatrix} \begin{bmatrix} A_0 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ A_K \end{bmatrix}$$

and

$$\begin{bmatrix} B(\xi_0) \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ B(\xi_K) \end{bmatrix} = \begin{bmatrix} \xi_0^0 \cdots \xi_0^K \\ \cdot \quad \cdot \\ \cdot \quad \cdot \\ \cdot \quad \cdot \\ \cdot \quad \cdot \\ \cdot \quad \cdot \\ \xi_K^0 \cdots \xi_K^K \end{bmatrix} \begin{bmatrix} B_0 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ B_K \end{bmatrix} \tag{5}$$

(here, $A_{J+1} = \cdots = A_K = 0$ and $B_{J+1} = \cdots = B_K = 0$ have been introduced to promote symmetry). The fourth step can also be expressed as the multiplication of a matrix by a vector,

$$\begin{bmatrix} C_0 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ C_K \end{bmatrix} = \begin{bmatrix} \eta_{0,0} \cdots \eta_{0,K} \\ \cdot \quad \cdot \\ \cdot \quad \cdot \\ \cdot \quad \cdot \\ \cdot \quad \cdot \\ \eta_{K,0} \cdots \eta_{K,K} \end{bmatrix} \begin{bmatrix} C(\xi_0) \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ C(\xi_K) \end{bmatrix}, \tag{6}$$

where the entries $\{\eta_{k,l}\}_{0 \le k \le K, 0 \le l \le K}$ depend upon the points $\{\xi_k\}_{0 \le k \le K}$ ($\eta_{k,l}$ is the coefficient of x^k in the polynomial $(x - \xi_0) \cdots (x - \xi_{l-1})(x - \xi_{l+1}) \cdots (x - \xi_K)/(\xi_l - \xi_0) \cdots (\xi_l - \xi_{l-1})(\xi_l - \xi_{l+1}) \cdots (\xi_l - \xi_K)$, which assumes the value 1 at ξ_l and the value 0 at $\xi_0, \cdots, \xi_{l-1}, \xi_{l+1}, \cdots, \xi_K$).

Although Winograd [7] has shown that the multiplication of an arbitrary $(K + 1) \times (K + 1)$ matrix by a $(K + 1)$-dimensional vector requires $(K + 1)^2$ multiplications, we are free to choose the points $\{\xi_k\}_{0 \le k \le K}$ so as to reduce this number if possible. A particularly favorable choice is to let $\{\xi_k\}_{0 \le k \le K}$ run through the $(K + 1)$st roots of unity. If $\xi_k = \zeta^k$, where ζ is a primitive $(K + 1)$st root of unity, then $\xi_k^l = \zeta^{kl}$, and the linear transformation in (5) is called the *Fourier transform*. Most marvelously, $\eta_{k,l} = \zeta^{-kl}$ and the linear transformation in (6) is called the *inverse Fourier transform*. (We are assuming here that the underlying field contains the $(K + 1)$st roots of unity. For simplicity we shall take it to be complex field **C**. Operations on complex numbers can be implemented by means of operations on their real and imaginary parts, so the results will apply with minor adjustments to the real field **R** as well.)

We turn now to the problem of computing the Fourier transform, where for notational convenience we take the number of points to be $N = K + 1$. There is of course an obvious algorithm requiring $O(N^2)$ operations. Good [16] showed that if $N = PQ$, where P and Q are relatively prime, then a Fourier transform on N points can be performed by means of P Fourier transforms on Q points and Q Fourier transforms on P points. Choosing N to be a product of distinct small primes yields an algorithm requiring only $O(N[\log N]^2/\log\log N)$ operations, though no one appears to have observed this. It was not until Cooley and Tukey [17] showed that if $N = 2^n$, then a Fourier transform can be performed with only $O(N \log N)$ operations that the ease of performing Fourier transforms was widely recognized.

The problem of computing the inverse Fourier transform is equivalent to that of computing the Fourier transform itself; the two transforms differ only in the replacement of ζ by ζ^{-1}. Thus the algorithm of Cooley and Tukey gives an algorithm for multiplying two polynomials of degree N requiring only $O(N \log N)$ operations. It is worth noting that only the $K + 1$ multiplications in the third step of the strategy are nonscalar multiplications; only additions and scalar multiplications are required to compute the Fourier transforms in the second step and the inverse Fourier transform in the fourth. Indeed, $p(T) = K + 1$ for the tensor T associated with the set of bilinear forms (3).

Further improved algorithms for the Fourier transform have been given by Winograd [18]. Winograd's algorithms, which include the best currently available, combine the ideas of Good [16] and Rader [19] with his own previous work [20] on the problem of multiplying two polynomials modulo a third.

The most penetrating study of the complexity of performing the Fourier transform,

$$\begin{bmatrix} G_0 \\ G_1 \\ \cdot \\ \cdot \\ \cdot \\ G_{N-1} \end{bmatrix} = \begin{bmatrix} \zeta^0 & \zeta^0 & \cdots & \zeta^0 \\ \zeta^0 & \zeta^1 & \cdots & \zeta^{N-1} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \zeta^0 & \zeta^{N-1} & \cdots & \zeta^{(N-1)^2} \end{bmatrix} \begin{bmatrix} F_0 \\ F_1 \\ \cdot \\ \cdot \\ \cdot \\ F_{N-1} \end{bmatrix},$$

arises by regarding $\{G_n\}_{0 \le n \le N-1}$ as a set of bilinear forms in the indeterminates $\{\zeta^n\}_{0 \le n \le N-1}$ and $\{F_n\}_{0 \le n \le N-1}$, where the indeterminates $\{\zeta^n\}_{0 \le n \le N-1}$ are no longer transcendental but satisfy certain algebraic relations. (We must now assume that the underlying field does *not* contain a primitive Nth root of unity. For simplicity, we shall take it to be the rational field **Q** in this paragraph.) Winograd [21], by extending the theory of bilinear forms to take account of these algebraic relations, has determined the number of "nonscalar multiplications" in this sense required to compute the Fourier transform when N is prime. Auslander and Winograd have recently extended this to arbitrary N.

The existence of fast algorithms for polynomial multiplication has implications for many other computational problems. It can, of course, be immediately adapted to give multiplication of (initial segments of) formal power series in $O(N \log N)$ operations. Sieveking [22] showed how to combine it with Newton's method of iteration to give reciprocation of power series in $O(N \log N)$ operations, and this in turn can be adapted to give polynomial division (with remainder) in $O(N \log N)$ operations.

Brent [23] extended Sieveking's method to the computation of square roots, exponentials, and logarithms of power series in $O(N \log N)$ operations and showed how this can be adapted to give iterated multiplication of power series in $O(N \log N)$ operations. Kung and Traub [24] showed that the power series expansions of all algebraic functions can be computed in $O(N \log N)$ operations. Brent and Kung [25] gave algorithms for composition and reversion of power series in $O([N \log N]^{3/2})$ operations, and Brent and Traub [26] showed how to combine this with Schröder's method of iteration to give iterated compositions of power series in $O([N \log N]^{3/2})$ operations.

Another class of problems that benefit from fast polynomial multiplication concern evaluation of polynomials at and interpolation of polynomials through an arbitrary set of N points (rather than Nth roots of unity, as for the Fourier transform). Moenck and Borodin [27] reduced these problems to polynomial multiplication and division; when Sieveking's method of polynomial division is used, the resulting algorithms require $O(N[\log N]^2)$ operations. These problems are among the few for which nonlinear lower bounds have been established. Strassen [28] (see also Schönhage [29]) has shown that evaluation of the polynomial $P(x) = x^N$ at N points and interpolation of a polynomial of degree N through N prescribed zeros each require at least $N \log_2 N + O(N)$ nonscalar multiplications and divisions. [The algorithms resulting from the reduction of Moenck and Borodin actually require only $O(N \log N)$ nonscalar multiplications and divisions; the larger bounds of $O(N[\log N]^2)$ include scalar multiplications and additions.] Gustavson and Yun [30] have extended the work of Moenck and Borodin to allow some of the N points to coincide, in which case the values of the polynomial and an appropriate number of its derivatives are to be computed (in the case of evaluation) or are prescribed (in the case of interpolation). The resulting algorithms again require $O(N[\log N]^2)$ operations.

Still another class of problems that benefit from fast-polynomial multiplication center around Euclid's algorithm for computing greatest common divisors. Moenck [31] has adapted this to exploit fast polynomial multiplication and division, obtaining an algorithm for the polynomial greatest common divisor (or least common multiple) requiring $O(N[\log N]^2)$ operations. Moenck's algorithm has been extended by Brent, Gustavson, and Yun [32] to yield as by-products information useful for performing interpolation of rational functions and their derivatives through an arbitrary set of points. This information is also useful for the solution of systems of linear equations in which the coefficients form Hankel or Toeplitz matrices. All of these computations can be done with $O(N[\log N]^2)$ operations.

Matrix multiplication

We now come to what many regard as the premier problem of algebraic complexity theory: matrix multiplication. Suppose that we are given an $I \times J$ matrix $\{A_{i,j}\}_{1 \le i \le I, 1 \le j \le J}$ and a $J \times K$ matrix $\{B_{j,k}\}_{1 \le j \le J, 1 \le k \le K}$ and that we want to compute their product, an $I \times K$ matrix $\{C_{i,k}\}_{1 \le i \le I, 1 \le k \le K}$ whose entries are given by the formulae

$$C_{i,k} = \sum_{1 \le j \le J} A_{i,j} B_{j,k} . \tag{7}$$

Let $T(I, J, K)$ denote the tensor associated with this set of bilinear forms.

We shall be particularly interested in the case $I = J = K = N$ of $N \times N$ square matrices. An obvious algorithm based on the formulae (7) requires N^3 multiplications and $N^2(N - 1)$ additions. In 1968, Winograd [6] gave an alternate algorithm requiring $N^3/2 + O(N^2)$ multiplications and $3N^3/2 + O(N^2)$ additions and subtractions. But the first reduction in the overall number of operations was obtained in 1969 by Strassen [33], who showed that $O(N^\alpha)$ operations suffice, where $\alpha = \log_2 7 = 2.807 \cdots$.

Strassen's reduction is based on two key facts. The first, which had been observed independently by Winograd [7], is that if $\mu^*(T(M, M, M)) \le M^\theta$ for some particular M and some $2 < \theta \le 3$, then $O(N^\theta)$ operations suffice for all N. The second is that $\mu^*(T(2, 2, 2)) \le 7$, which was shown by means of an explicit algorithm for multiplying 2×2 matrices.

Hopcroft and Kerr [5] have shown that $\mu^*(T(2, 2, 2)) = 7$ [indeed, Winograd [1] has shown that $\mu(T(2, 2, 2)) = 7$], so Strassen's exponent cannot be reduced through further consideration of 2×2 matrices. Strassen's algorithm achieving $\mu^*(T(2, 2, 2)) = 7$ used 18 additions and subtractions; Winograd (see Probert [34]) reduced this to 15, and Probert [34] showed that 15 is minimal. This reduction affects only the constant factor implicit in $O(N^\alpha)$, however, and not the exponent α.

After the resources of the case $M = 2$ were exhausted, it was natural to hope that Strassen's exponent might be reduced by showing that $\mu^*(T(3, 3, 3)) \leq 21$, $\mu^*(T(4, 4, 4)) \leq 48$, or $\mu^*(T(5, 5, 5)) \leq 91$. Despite many attempts along these lines (see for example Laderman [35] and Schachtel [36]), no progress was made in reducing the exponent until Pan [37] showed by an ingenious algorithm that $\mu^*(T(M, M, M)) \leq M^3/3 + 9M^2/2 - M/3$, which yields $\mu^*(T(48, 48, 48)) \leq 47\,216$, and thus that $O(N^\beta)$ operations suffice for $\beta = \log_{48} 47\,216 = 2.780 \cdots$.

What is perhaps the most striking contribution to the problem of matrix multiplication, however, stems from the following observation. For matrices S, the condition $\rho(S) \leq r$ that S have rank at most r can be expressed in terms of the vanishing of certain determinants. As a consequence, if $\rho(S^{(n)}) \leq r$ holds for all matrices in a sequence $S^{(1)}, S^{(2)}, \cdots$ converging to S, then (since determinants are continuous functions of the entries of matrices) $\rho(S) \leq r$ holds as well. Contrastingly, the rank of a tensor T need not be continuous in this way. It is possible to have $\rho(T^{(n)}) \leq r$ for $T^{(1)}, T^{(2)}, \cdots$ converging to T, but at the same time to have $\rho(T) \geq r + 1$. The significance of this for computational complexity is that it may require fewer operations to compute a set of bilinear forms with arbitrarily small error than to compute them exactly.

An example of this phenomenon, due to Schönhage [38], is the following. Consider the problem of computing the bilinear forms

$$C_1 = A_1B_1 ,$$

$$C_2 = A_1B_2 + A_2B_1 .$$

If T denotes the associated tensor, it is not hard to show that $\mu^*(T) = \rho(T) = 3$. But if we compute

$$M_1 = A_1B_1 ,$$

$$M_2 = (A_1 + \epsilon A_2)(B_1 + \epsilon B_2) ,$$

then

$$C_1 = M_1 ,$$

$$C_2 = \epsilon^{-1}(M_2 - M_1) - \epsilon A_2B_2 .$$

Thus, with only two nonscalar multiplications, C_1 and C_2 can be computed with an error that can be made as small as desired by choosing ϵ small enough. Schönhage defined a new rank $\rho_0(T)$, called the *border rank*, of a tensor T for which the minimum number of nonscalar multiplications $\mu_0^*(T)$ sufficient to compute the associated set of bilinear forms in noncommuting indeterminates with arbitrarily small error satisfies the identity $\mu_0^*(T) = \rho_0(T)$. In the example just given, $\mu_0^*(T) = \rho_0(T) = 2$.

The case of performing approximate computations can often be exploited in performing larger but exact computations. This is done by performing the approximate computations in the field of formal power series in the indeterminate ϵ, using Fourier transforms to expedite the operations on power series as described in the preceding section. For matrix multiplication, Bini [39], Romani [40], and Winograd (see Pan [41]) have shown that if $\mu_0^*(T(M, M, M)) \leq M^\theta$ for some particular M and some $2 < \theta \leq 3$, then $O(N^\theta \log N)$ operations suffice for exact matrix multiplication.

By the duality principle, $\mu^*(I, J, K) = \mu^*(J, K, I) = \mu^*(K, I, J)$. Furthermore, $T(IJK, IJK, IJK) = T(I, J, K) \otimes T(J, K, I) \otimes T(K, I, J)$, where $T \otimes T'$ denotes what is called the *tensor product* of T and T'. In general, $\mu^*(T \otimes T') \leq \mu^*(T)\mu^*(T')$, and so $\mu^*(T(IJK, IJK, IJK)) \leq \mu^*(T(I, J, K))^3$. Thus, if $\mu^*(T(I, J, K)) \leq (IJK)^{\theta/3}$ for some particular I, J, and K and some $2 < \theta \leq 3$, then $O(N^\theta)$ operations suffice for matrix multiplication. The duality principle and the tensor product inequality apply to μ_0^* as well as μ^*, and thus if $\mu_0^*(T(I, J, K)) \leq (IJK)^{\theta/3}$ for some particular I, J, and K and some $2 < \theta \leq 3$, then $O(N^\theta \log N)$ operations suffice for exact matrix multiplication. In 1979, Bini, Capovani, Romani, and Lotti [42] showed that $\mu_0^*(T(3, 2, 2)) \leq 10$, which yields that $O(N^\gamma \log N)$ operations suffice for $\gamma = 3 \log_{12} 10 = 2.779 \cdots$.

Still further reductions in the exponent have been obtained by Pan and Schönhage (see Schönhage [38]). They showed that if $\mu^*(T(I_1, J_1, K_1) \oplus \cdots \oplus T(I_s, J_s, K_s)) \leq (I_1J_1K_1)^{\theta/3} \oplus \cdots \oplus (I_sJ_sK_s)^{\theta/3}$ for some particular I_1, J_1, K_1, $\cdots$, I_s, J_s, K_s and some $2 < \theta \leq 3$, then $O(N^\theta)$ operations suffice. A similar result holds with μ^* replaced by μ_0^* and $O(N^\theta)$ replaced by $O(N^\theta \log N)$. They have also shown that $\mu_0^*(T(1, 5, 22) \oplus T(11, 2, 5) \oplus T(10, 11, 1)) \leq 156$, which yields that $O(N^\delta \log N)$ operations suffice, where $\delta = 3 \log_{1110} 52 = 2.521 \cdots$. Coppersmith and Winograd have recently used this method and extensions of it to reduce the exponent still further, to $2.495 \cdots$.

The existence of fast algorithms for matrix multiplication has implications for many other problems of linear algebra. As was observed by Strassen [33], if matrix multiplication can be performed with $O(N^\theta)$ operations, for some $2 < \theta \leq 3$, so can matrix inversion; the converse result has been obtained by Munro [43]. Other problems that can similarly be reduced to matrix multiplication are solution of systems of linear equations, triangular factorization and evaluation of determinants (see Bunch and Hopcroft [44]), and orthogonalization (see Schönhage [45]).

Conclusion

In the preceding section, we tacitly assumed that "faster" means "better," but time is often not the only or even the most important resource to be reckoned in assessing the complexity of computations. The other resource most frequently mentioned is space, the maximum number of intermediate results that need to be kept at any point in the execution of an algorithm. Of particular interest are time-space tradeoffs: situations in which minimum time and minimum space cannot be achieved by the same algorithm and in which a spectrum of algorithms, each optimal according to its own objective, therefore exists. The first results along these lines were obtained by Grigorev [46]; for others see Tompa [47] and Ja' Ja' [48].

Another resource often discussed is depth, which can be described as parallel time (the time required when any number of operations may be performed at once), in contrast to serial time (the time required when operations must be performed one after another, referred to simply as "time" above). Depth was given a status equal to that of time by Strassen [49, 50] in his formulation of algebraic complexity. Some important upper bounds to depth are those for reciprocation of power series (implicit in Sieveking [22]), for inversion of matrices (due to Csanky [51]), and for arbitrary polynomials of limited degree that are computable in limited time (due to Hyafil [52]). It is an open problem to obtain a lower bound to depth growing faster than the logarithm of the number of input data for any computation.

Although nonlinear lower bounds in algebraic complexity theory are scarce, there is one situation in which they have been obtained with relative ease: that in which algorithms are assumed to satisfy some strong restriction. The most common such restriction is to *monotone* algorithms, which may use positive but not negative real constants, additions but not subtractions, and multiplications but not divisions. Kerr [53] showed that about N^3 operations are necessary in a monotone algorithm for multiplying $N \times N$ matrices, and Schnorr [54] has shown that about N^2 operations are necessary in a monotone algorithm for multiplying polynomials of degree N. Since, as we have seen in the two preceding sections, faster nonmonotone algorithms for these problems exist, these results show something of the power of nonmonotonicity. Schnorr [54] and also Shamir and Snir [55] have obtained lower bounds that grow exponentially with the number of input data, though no significantly faster nonmonotone algorithms are known for the problems they treat. Valiant [56], however, has given an example showing that nonmonotonicity can reduce the complexity of a problem from an exponential to a polynomial in the number of input data.

References

1. S. Winograd, *Linear Algebra Appl.* **4,** 381–388 (1971).
2. H. F. de Groote, *Info. Proc. Lett.* **3,** 177–179 (1975).
3. S. Winograd, *Actes Congress Intern. Math.* **3,** 283–288 (1970).
4. V. Strassen, *J. Reine Angew. Math.* **264,** 184–202 (1973).
5. J. E. Hopcroft and L. R. Kerr, *SIAM J. Appl. Math.* **20,** 30–36 (1971).
6. S. Winograd, *IEEE Trans. Computers* **C-17,** 693–694 (1968).
7. S. Winograd, *Commun. Pure Appl. Math.* **23,** 165–179 (1970); preliminary version: *Proc. Nat. Acad. Sci. USA* **58,** 1840–1842 (1967).
8. J. Ja' Ja', *ACM Symp. Theor. Comput.* **11,** 197–208 (1979).
9. L. Hyafil, *IEEE Symp. Found. Comput. Sci.* **18,** 171–174 (1977).
10. N. Gastinel, *Num. Math.* **17,** 222–229 (1971).
11. C. M. Fiduccia, *Complexity of Computer Computations*, Plenum Press, Inc., New York, 1972, pp. 33–40.
12. V. Strassen, *Complexity of Computer Computations*, Plenum Press, Inc., New York, 1972, pp. 1–10.
13. D. Yu. Grigorev, *Investigations on Linear Operations*, Nauka, 1974, pp. 159–163.
14. J. Ja' Ja', *SIAM J. Computing* **8,** 443–462 (1979).
15. J. Ja' Ja', *SIAM J. Appl. Math.* **37,** 700–712 (1979).
16. I. J. Good, *J. Roy. Stat. Soc. B* **20,** 361–372 (1958); addendum; **22,** 372–375 (1960).
17. J. W. Cooley and J. W. Tukey, *Math. Comput.* **19,** 297–301 (1965).
18. S. Winograd, *Math. Comp.* **32,** 175–199 (1978); preliminary version: *Proc. Nat. Acad. Sci. USA* **73,** 1005–1006 (1976).
19. C. M. Rader, *Proc. IEEE* **5,** 1102–1108 (1968).
20. S. Winograd, *Math. Syst. Theor.* **10,** 169–180 (1977); preliminary version: *IEEE Symp. Found. Comput. Sci.* **16** (1975). See also S. Winograd, *Theor. Comput. Sci.* **8,** 359–377 (1979); S. Winograd, *SIAM J. Computing* **9,** 225–229 (1980).
21. S. Winograd, *Adv. Math.* **32,** 83–117 (1979).
22. M. Sieveking, *Computing* **10,** 153–156 (1972).
23. R. P. Brent, *J. ACM* **23,** 242–251 (1976).
24. H. T. Kung and J. F. Traub, *J. ACM* **25,** 245–260 (1978).
25. R. P. Brent and H. T. Kung, *J. ACM* **25,** 581–595 (1978).
26. R. P. Brent and J. F. Traub, *SIAM J. Computing* **9,** 54–66 (1980).
27. R. Moenck and A. Borodin, *IEEE Symp. on Switching and Automation Theory* **13,** 90–96 (1972).
28. V. Strassen, *Num. Math.* **20,** 238–251 (1973).
29. A. Schönhage, *Theor. Comput. Sci.* **3,** 267–272 (1976).
30. F. G. Gustavson and D. Y. Y. Yun, *IEEE Trans. Circuits Syst.* **26,** 750–755 (1979).
31. R. T. Moenck, *ACM Symp. Theor. Comput.* **5,** 142–151 (1973).
32. R. P. Brent, F. G. Gustavson, and D. Y. Y. Yun, *J. Algorithms* **1,** 259–295 (1980).
33. V. Strassen, *Num. Math.* **13,** 354–356 (1969).
34. R. L. Probert, *SIAM J. Computing* **5,** 187–203 (1976).
35. J. D. Laderman, *Bull. AMS* **82,** 126–128 (1976).
36. G. Schachtel, *Info. Proc. Lett.* **7,** 180–182 (1978).
37. V. Ya. Pan, *SIAM J. Computing* **9,** 321–342 (1980); preliminary version: *IEEE Symp. Found. Comput. Sci.* **19,** 166–176 (1978).
38. A. Schönhage, *Technical Report on Partial and Total Matrix Multiplication*, Mathematisches Institut, Universität Tübinque, June 1979; revised January 1980.
39. D. Bini, *Nota Interna B79/8*, Istituto di Elaborazione dell'Informazione, Pisa, Italy, March 1979.
40. F. Romani, *Nota Interna B79/9*, Istituto di Elaborazione dell'Informazione, Pisa, Italy, April 1979.
41. V. Ya. Pan, *IEEE Symp. Found. Comput. Sci.* **20,** 28–38 (1979).
42. D. Bini, M. Capovani, F. Romani, and G. Lotti, *Info. Proc. Lett.* **8,** 234–235 (1979).
43. J. I. Munro, *Courant Inst. Symp. on Computational Complexity*, Algorithmic Press, 1973, pp. 137–152.

44. J. Bunch and J. E. Hopcroft, *Math. Comp.* **28,** 231–236 (1974).
45. A. Schönhage *Complexity of Sequential and Parallel Numerical Algorithms*, Academic Press, Inc., New York, 1973, pp. 283–291.
46. D. Yu. Grigorev, *Issled. po Konstr. Mat. i Mat. Log.*, 38–48 (1976).
47. M. Tompa, *J. Computer Syst. Sci.* **20,** 118–132 (1980); preliminary version: *ACM Symp. Theor. Comput.* **10,** 196–204 (1978).
48. J. Ja' Ja', *ACM Symp. Theor. Comput.* **12,** 339–350 (1980).
49. V. Strassen, *Acta Informatica* **1,** 320–335 (1972).
50. V. Strassen, *Acta Informatica* **2,** 64–79 (1973).
51. L. Csanky, *SIAM J. Computing* **5,** 618–623 (1976); preliminary version: *IEEE Symp. Found. Comput. Sci.* **16,** 11–12 (1975).
52. L. Hyafil, *SIAM J. Computing* **8,** 120–123 (1979); preliminary version: *ACM Symp. Theor. Comput.* **10,** 193–195 (1978).
53. L. R. Kerr, *The Effect of Algebraic Structure on the Computational Complexity of Matrix Multiplication*, Ph.D. Thesis, Cornell University, Ithaca, NY, 1970.
54. C. P. Schnorr, *Theor. Comp. Sci.* **2,** 305–315 (1976).
55. E. Shamir and M. Snir, *IBM Research Report RC-6757*, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, 1977.
56. L. G. Valiant, *ACM Symp. Theor. Comput.* **11,** 189–196 (1979).

Received October 8, 1980; revised February 12, 1981

The author is located at the IBM Research Division laboratory, 5600 Cottle Road, San Jose, California 95193.